/\*

HCSR04 - Library for arduino, for HC-SR04 ultrasonic distance sensor.

Created by Dirk Sarodnick, 2020.

\*/

#ifndef HCSR04\_H

#define HCSR04\_H

#include "Arduino.h"

#define HCSR04\_INVALID\_RESULT -1;

#define HCSR04\_NO\_TRIGGER -2;

#define HCSR04\_NO\_ECHO -3;

class HCSR04Sensor {

public:

HCSR04Sensor();

~HCSR04Sensor();

typedef enum eUltraSonicUnlock {

unlockSkip = 0,

unlockMaybe = 1,

unlockForced = 2

} eUltraSonicUnlock\_t;

void begin(uint8\_t triggerPin, uint8\_t echoPin) { begin(triggerPin, new uint8\_t[1]{ echoPin }, 1); }

void begin(uint8\_t triggerPin, uint8\_t\* echoPins, uint8\_t echoCount) { begin(triggerPin, echoPins, echoCount, 100000, eUltraSonicUnlock\_t::unlockSkip); }

void begin(uint8\_t triggerPin, uint8\_t echoPin, uint32\_t timeout, eUltraSonicUnlock\_t unlock) { begin(triggerPin, new uint8\_t[1]{ echoPin }, 1, timeout, unlock); }

void begin(uint8\_t triggerPin, uint8\_t\* echoPins, uint8\_t echoCount, uint32\_t timeout, eUltraSonicUnlock\_t unlock) { begin(triggerPin, echoPins, echoCount, timeout, 10, 10, unlock); }

void begin(uint8\_t triggerPin, uint8\_t\* echoPins, uint8\_t echoCount, uint32\_t timeout, uint16\_t triggerTime, uint16\_t triggerWait, eUltraSonicUnlock\_t unlock);

void end();

long\* measureMicroseconds() { measureMicroseconds(lastMicroseconds); return lastMicroseconds; }

void measureMicroseconds(long\* results);

double\* measureDistanceMm() { measureDistanceMm(defaultTemperature, lastDistances); return lastDistances; }

void measureDistanceMm(double\* results) { measureDistanceMm(defaultTemperature, results == NULL ? lastDistances : results); }

double\* measureDistanceMm(float temperature) { measureDistanceMm(temperature, lastDistances); return lastDistances; }

void measureDistanceMm(float temperature, double\* results);

double\* measureDistanceCm() { measureDistanceCm(defaultTemperature, lastDistances); return lastDistances; }

void measureDistanceCm(double\* results) { measureDistanceCm(defaultTemperature, results == NULL ? lastDistances : results); }

double\* measureDistanceCm(float temperature) { measureDistanceCm(temperature, lastDistances); return lastDistances; }

void measureDistanceCm(float temperature, double\* results);

double\* measureDistanceIn() { measureDistanceIn(defaultTemperature, lastDistances); return lastDistances; }

void measureDistanceIn(double\* results) { measureDistanceIn(defaultTemperature, results == NULL ? lastDistances : results); }

double\* measureDistanceIn(float temperature) { measureDistanceIn(temperature, lastDistances); return lastDistances; }

void measureDistanceIn(float temperature, double\* results);

static void triggerInterrupt0(void);

static void triggerInterrupt1(void);

static void triggerInterrupt2(void);

static void triggerInterrupt3(void);

static void triggerInterrupt4(void);

static void triggerInterrupt5(void);

static void triggerInterrupt6(void);

static void triggerInterrupt7(void);

static void triggerInterrupt8(void);

static void triggerInterrupt9(void);

static void echoInterrupt0(void);

static void echoInterrupt1(void);

static void echoInterrupt2(void);

static void echoInterrupt3(void);

static void echoInterrupt4(void);

static void echoInterrupt5(void);

static void echoInterrupt6(void);

static void echoInterrupt7(void);

static void echoInterrupt8(void);

static void echoInterrupt9(void);

private:

float defaultTemperature = 19.307;

long\* lastMicroseconds;

double\* lastDistances;

uint32\_t timeout;

uint16\_t triggerTime = 10; // HC-SR04 needs at least 10µs trigger. Others may need longer trigger pulses.

uint16\_t triggerWait = 10; // HC-SR04 sends its signal about 200µs. We only wait a small amount to reduce interference, but to not miss anything on slower clock speeds.

volatile uint8\_t triggerPin;

volatile unsigned long\* volatile triggerTimes;

uint8\_t echoCount;

volatile int16\_t\* volatile echoStages;

volatile int16\_t\* volatile echoInts;

volatile int16\_t\* volatile echoPorts;

volatile unsigned long\* volatile echoTimes;

void triggerInterrupt(uint8\_t);

void echoInterrupt(uint8\_t);

void unlockSensors(eUltraSonicUnlock\_t, uint8\_t\*);

};

extern HCSR04Sensor HCSR04;

#endif // HCSR04\_H